REMARKS

The following is intended as a full and complete response to the Office Action dated June 4, 2008. Claims 1, 2, 4-8, and 10-20 were examined. Claims 1, 2, 4-8, 10-14, and 22-29 are rejected under 35 U.S.C. § 102(a) as being anticipated by Elzur (US 7,346,701). Claims 15-21 and 30 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Elzur in view of Odenwald (US 6,310,884), claims 22-30 are described as being rejected under 35 U.S.C. § 103(a) as being unpatentable over Elzur in view of Odenwald as well.

Claims 1, 2, 4-8, 10-14, and 29

As amended, claim 1 recites the limitations of selecting, by a transmission control protocol (TCP) stack, a connection for processing by an offload unit and storing a sequence number from the first frame in an entry of the delegated connection table. As supported in paragraph [0045] of the present application, the TCP stack selects one or more connections as delegated connections that are processed by an offload unit. As supported in Figure 8A and paragraph [0100] of the present application, an offload unit is configured to store frame sequence numbers.

The Elzur reference fails to teach each and every limitation recited in amended claim 1. The TCP offload apparatus of Elzur processes all TCP packets, buffers portions of the packets in elastic buffers, and places portions of the packets in host memory. Nowhere does Elzur teach or suggest that a connection is selected by a TCP stack for offload processing, as claimed. In contrast, as the reference makes clear, each and every incoming TCP packet is processed to some extent by the TEEC. In addition, Elzur teaches that context information is stored in a memory and accessed to process frames for a connection (see col. 10, lines 46-48 and lines 59-65 and Figures 12, 13, and 14). However, Elzur does not describe the specific connection state that is included in the context information. In particular, Elzur fails to teach or suggest the limitation of storing a sequence number from the first frame in an entry of the delegated connection table, as claimed. Since Elzur fails to teach each and every limitation recited in amended claim 1, this claim and dependent claims 2, 4-8, 10-14, and 29 cannot be anticipated by Elzur.

Further, claim 2 recites the limitation of copying a portion of the second frame into a portion of the entry in the delegated connection table. Column 15, lines 1-7 of Elzur specifically teaches "copying data of an incoming packet to a host resident buffer…". The host

resident buffer is located in the host memory and is not equivalent to an entry of a delegated connection table. Therefore, Elzur fails to teach or suggest the limitations of claim 2 as well.

Claim 5 is amended to clarify that a legacy buffer is in a portion of the memory that is allocated to the driver. Elzur teaches uploading TCP payload data to host buffers in host memory (see Figures 2, 9, and step 170 of Figure 11), but fails to teach that the host buffers are in a portion of memory allocated to a driver. In contrast, in column 15, lines 30-34, Elzur expressly teaches that the host buffers may be pre-posted application buffer. Nowhere does Elzur teach that the host buffers are associated with a driver. Amended claim 5 also recites the limitation of notifying the TCP stack when payload data is uploaded. Elzur fails to teach or suggest this limitation too. The Examiner states that Elzur describes a notification in step 130 of Figure 11 and column 14, lines 10-19. However, step 130 teaches fetching connection context and does not relate to any type of notification.

Claim 8 is amended to recite the limitations of determining that the first frame and the second frame are out-of-sequence based on comparing the sequence number stored in the entry with a sequence number in the second frame, and storing a flag in the entry to indicate that synchronization is requested for the connection. These limitations are supported by step 812 and paragraph [0099] of the present application. As previously explained, Elzur fails to teach storing sequence numbers. Elzur also fails to teach comparing sequence numbers and storing a synchronization flag in the entry based on that comparison.

Claim 10 recites the limitation that the payload data is uploaded to the legacy buffer when a user buffer is not available. The legacy buffer is in a portion of memory allocated to a driver, and the user buffer is in a portion of memory that is allocated to an application program. In contrast, Elzur fails to teach or suggest that payload data is uploaded to buffers in portions of memory that are allocated to different clients.

Claim 12 recites the limitations of uploading any subsequent frames received for the connection to one or more additional legacy buffers until resynchronization is signaled by the TCP stack. Elzur fails to teach any type of resynchronization. The Examiner states that the limitations of claim 12 are described in lines 4-9, without specifying a column number.

Claims 13 and 14 are amended to recite the limitations of invalidating any buffer descriptors for portions of the memory that are available for storing data received on the connection. These amendments are supported by paragraph [0103] of the present application. Nowhere does Elzur teach invalidating host buffers or any buffer descriptors.

Claim 29 recites the limitations of updating connection state data that includes clearing an unACKnowledged count, updating the ACK number with a last ACKnowledged number, and updating the expected sequence number with an incremental sequence number. These limitations are described in [00145] of the present application. Elzur fails to teach or suggest clearing an unACKnowledged count, updating the ACK number with a last ACKnowledged number, and updating the expected sequence number with an incremental sequence number. As previously explained with regard to amended claim 1, Elzur fails to describe the specific connection state that is included in the context information. The reference simply doesn't disclose anything aout storing an unACKnowledged count, an ACK number, or an expected sequence number.

Claims 15-28 and 30-31

Amended claims 15 and 22 recite the limitation that the delegated connections are selected by a transmission control protocol (TCP) stack for processing by an offload unit that includes the delegated connection table.

As previously described with regard to amended claim 1, Elzur fails to teach or suggest the limitation of a TCP stack selecting the delegated connections. Odenwald teaches only a method of transmitting data blocks to improve buffer allocation within the receiver and therefore fails to cure the deficiencies of Elzur relative to amended claims 15 and 22. For these reasons, no combination of Elzur and Odenwald can render amended claims 15 and 22 or dependent claims 16-21, 24-28, and 30-31 obvious.

In addition to the foregoing, claim 18 recites the limitations of setting a request buffer flag in the delegated connection table when a user buffer is not available. Elzur and Odenwald each fail to teach or suggest these limitations.

Claim 20 recites the limitations of determining that a receive buffer has reached a high water mark, and Claim 21 recites the limitations of determining that a buffer request timer has expired. Nowhere does Elzur teach or suggest the use of a high water mark or a buffer request timer. Odenwald also fails to teach or suggest these limitations.

Claim 30 recites the limitations of modifying connection state data that includes clearing an unACKnowledged count, updating the ACK number with a last ACKnowledged number, and updating the expected sequence number with an incremental sequence number. These limitations are described in [00145] of the present application. As previously explained with regard to amended claim 1, Elzur fails to describe the specific connection state that is

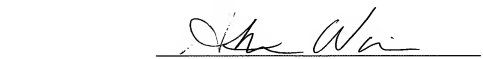
included in the context information. Figure 4 illustrates the fields of a TCP header, but the figure does not show an expected sequence number, timestamp data, or a count of unACKnowledged frames. More importantly, nowhere does Elzur teach or suggest storing TCP headers in a delegated connection table. Odenwald teaches receiving a count of frames, SEQ_CNT. Nowhere does Odenwald teach or suggest storing the count. In Figure 5 Odenwald shows an acknowledgement number, but Odenwald fails to teach an unACKnowledged count or storing the acknowledgement number. The illustration of a protocol header does not rise to the level of a teaching required to anticipate or render obvious the pending claims. Therefore, the combination of these two references fails to disclose the limitations of claim 30.

New claim 31 depends from amended claim 22 and recites the limitations of the delegated connection table storing delegated connections that are a subset of active connections stored in a connection table within a system memory. These limitations are supported by Figure 2B and paragraph [0046] of the present application. Neither Elzur nor Odenwald teach or suggest the limitation of a delegated connection table within the offload unit that stores a subset of the active connections stored in a connection table in the system memory. New claim 31 is therefore allowable over the cited references.

CONCLUSION

Based on the above remarks, Applicants believe that they have overcome all of the rejections set forth in the Office Action mailed on June 4, 2008 and that the pending claims are in condition for allowance. If the Examiner has any questions, please contact the Applicant's undersigned representative at the number provided below.

Respectfully submitted,



Stephanie Winner
Registration No. 52,371
PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Agent for Applicants